FINAL REPORT
Robust Interaction and Natural Problem Solving in Advice-Giving Systems
NSF Grant IST-8608362

Principal Investigators: R. E. Cullingford and J. L. Kolodner
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280

February 7, 1992

The intent of this project was to develop fundamental design principles for intelligent advisory systems that interact with their users in natural language during problem solving episodes. Three major integration problems were addressed in this context: control of conversation in an advisory system, control of problem solving processes in an interactive sytem, and integrating reactive and opportunistic reasoning processes with planful behavior. These problems were addressed within the framework provided by case-based reasoning, which also provided the vehicle for integrating their solutions. In addition, research on these issues led to two additional efforts: investigation of the integration of memory and reasoning processes and an exploration of case-based design. The first grew out of using case-based reasoning as the framework for this research. In case-based reasoning, a reasoner solves new problems by recalling old similar situations and adapting their solutions to the new problem. In both of the integrative efforts above, the reasoners used a combination of cases and generalized cases to do their job. We believed an investigation of the integration of these two types of knowledge in reasoning was in order. The investigation of case-based design grew out of the advisory system's domain: meal planning. Meal planning was treated as a design task. As such, it is of limited but interesting complexity, and this investigation has told us much about the cognition involved in design. The research was conducted primarily within the context of the JULIA system (Cullingford & Kolodner, 1986, Cullingford & Turner, 1988, Hinrichs, 1988, 1989, Kolodner, 1987, Shinn, 1988, Turner & Cullingford, 1988, 1989). JULIA is an advisory system whose task domain is catering and menu planning. Work on problem solving control was done in the context of the MEDIC project (Turner, 1988, 1989), whose task domain is diagnosis of pulmonary disorders. Four Ph.D. theses resulted from this research.

Control of conversation in an interactive advisory system: Elise Turner's thesis, "Integrating Intention and Convention To Organize Problem Solving Dialogues," reports on the control of conversation in an advisory system. Her system, JUDIS, achieves this through the use of a representation for conversation, called *conversation MOPs*, that combines intention and

convention. The structure of problem solving dialogues reflects both the conventions of language and the intentions, or goals, of conversants. Conversation MOPs capture both. The goals of a conversant, its intentions, derive from two places: demands of language mechanisms and demands of the reasoner or problem solver. Conventions of conversation derive from the culture as well as from the reasoning task underlying the conversation (e.g., clients and advisors have a certain way of interacting). Conversation is controlled by a mechanism that uses the conventions captured by conversation MOPs to create a plan for the conversation that will achieve the system's goals in a way that seems natural to the user. The planner is flexible enough to react to the changing goals of the system as well as to many types of unanticipated user utterances. The system achieves flexibility because it dynamically alters the plan as new goals arrive and because it combines the effects of intention and convention to make decisions about plan execution while participating in the conversation. Because goals are fit into the conventional framework of the accepted conversation MOPs, the natural language interface can organize even an incoherent stream of goals from the problem solver to form a coherent dialogue. Because the system allows a goal's priority to influence the actual execution of the plan, it can override convention to respond to important goals ina timely manner. Because intention and convention are combined, the resulting dialogue is both easy to understand and responsive to the needs of the problem solver.

Control of problem solving in an interactive reasoning system When a reasoner solves problems by interacting with another reasoner, its knowledge is incomplete for a number of reasons. (see Hinrichs (1988) for more detail and examples).

1. One of the roles of an advisory system is to help the user define his or her problem precisely. Thus, problems start out being under-constrained. This is a feature of cooperative problem solving systems in general. In JULIA, this problem continues throughout problem solving. Problems remain underconstrained since JULIA's domain is one where there is no single right answer and no best answer.

2. Not only is all the information not available from the start, but because of the complexity of the domain, neither the computer program nor the user might realize that some piece of information is important until late in the problem solving. This is typical of open-world problems.

3. Because the computer and the user are solving a problem cooperatively, the user might interrupt the computer at any time with new information or suggestions for how to proceed. The computer must be able to deal with interruptions both reactively and opportunistically.

4. In some domains, JULIA's included, decision criteria are subjective and idiosyncratic. This is particularly true in domains where aesthetics or other kinds of judgements play a large role in reasoning. An advisory system in such a domain must deal with the preferences of individual users. Some assumptions it makes about preferences or aesthetics may not correspond to those of its client user. And some may not come up until late in the problem solving. JULIA shares this complexity with many design domains (e.g., landscape design, display design, architecture).

5. The scope of problems in a cooperative problem solving system is ill-defined. Should the system solve the problem itself? At what points should it communicate with the user? How

much and which details should be presented when it does communicate? How does it know when it is done? A suggestion to one person can be fairly abstract and the person will know how to carry it out, while another person might need a far more detailed plan. The scope of the problem to be solved must be determined from context.

6. Some domain theories are necessarily incomplete. A system solving problems in such a domain cannot know everything.

Our approach to these problems has been to come up with an architecture that combines several problem solving approaches to be flexible enough to deal with all of the problems above. This requires integration of processes along two dimensions: First, we need to integrate processes that can postpone commitment to an answer until more information is available (least-commitment processes) with processes that can make assumptions or educated guesses about missing information (immediate-commitment processes). In JULIA, we use a constraint propagator for the first task and a case-based reasoner for the second. Second, we need to integrate reactive and opportunistic problem solving processes together. Reactive processes react to new unexpected information from the environment. Opportunistic processes keep track of what they need to know or find out so that they can recognize opportunities to use new information as it comes in. Additional detail on both of these solutions is in the next few sections.

Integrating reactive and opportunistic reasoning processes with planful behavior: In advisory systems in particular, and open-world problem solving in general, one can't foresee everything that will happen at the start of problem solving. Thus, it is necessary for the reasoning system to be able to do some amount of pre-planning but to be able to defer planning of details until execution time and to be able to interrupt itself as unexpected events take place. MEDIC implements a technique called *schema-based reasoning* (SBR) that supports this.

The SBR approach to problem solving combines case-based reasoning with reactive control, resulting in a reasoner that can both react appropriately to changes in its environment and be opportunistic. Using a hybrid of case-based reasoning in which abstract cases as well as real cases are used, and case parts as well as full cases are used, the problem solver plans its activity based on its previous experience rather than having to derive its entire plan from scratch each time. Case-based reasoning is not sufficient, however, for the entire problem solving task because new unexpected information or changes in the environment may crop up at any time. While problem solver should rely on its previous experience to plan its activities, it should not so bound to its previous plans that it can't adapt to the new situation. In SBR, this is solved by integrating planning with plan execution. The problem solver creates a partial plan and begins to carry it out. It adds to the plan as it learns more and it adapts the plan as it gets feedback or unexpected information. The goal scheduler that is part of the architecture knows how to interleave parts of procedures with parts of other procedures. This approach results in execution-time flexibility, and allows the planner to be opportunistic.

The approach shares much with Georgeff and Lansky's (1987) and Firby's (1987) approaches to combining reactive control with planning, but is novel in several important ways. First, we have derived a representation for procedures that takes into account the sequencing of procedure steps and the conditions under which the procedure and its steps can be done. Procedures are represented in a goal-oriented way. That is, there are a set of goals that must be achieved by a procedure. They

are achieved through actions. Sometimes those actions can be specified completely and sometimes only abstractly. Each step knows which goal(s) it achieves.

Second, the organization and access of procedures is such that procedures are represented at several different levels of abstraction. At any point in time, the most specific applicable procedure is retrieved and used. The organization and access procedures are based on MOPs (Schank, 1982, Kolodner, 1984).

Third, we have defined three kinds of knowledge structures and the contents they need to have in order to make reasoning flexible. Contextual schemata describe situations in a descriptive way and keep track of which kinds of goals are usually active in those situations. They also know what kinds of priorities goals have in the situation they define. Procedural schemata know the subgoals and/or steps (actions) involved in achieving any goal. Strategic schemata hold information about how how to control the application of procedures. Schemata are indexed in the memory by features that predict their applicability.

The result of all of this is a flexible control scheme combining planning and execution that reuses old reasoning whenever possible, thus avoiding a lot of planning from scratch; preplans to an appropriate level of detail, deferring details until execution time; reacts to important changes in its environment; and takes advantage of opportunities when they arise. This work has stretched the case-based reasoning paradigm significantly, and at the same time, has added to work done on reactive control and opportunistic reasoning. The view of problem solving, combining planning and execution, is a break with old paradigms of planning and fits squarely with newly-emerging planning paradigms.

Roy Turner's thesis, "A Schema-Based Model of Adaptive Problem Solving," reports on this effort. This model of adaptive reasoning is implemented in MEDIC, a schema-based diagnostic consultant in the domain of pulmonology. MEDIC diagnoses simple cases of purmonary disease by using a version of the INTERNIST-1 diagnostic algorithm, which is implemented as procedural schemas.

Integrating memory and reasoning: The approach to case-based reasoning being studied under this grant is called *abstractional analogy* (Shinn, 1988). Abstractional analogy has five steps:

1. Map the old and new problems to each other, finding correspondences between the problems.
2. Create a *problem schema* that describes the commonalities between the two problems.
3. Create a *solution schema* from the previous solution that is at the same level of detail as the problem schema and that describes how the goals of the problem schema were achieved.
4. Apply the solution schema to the new problem to create a partial solution, analogous to the previous solution at the level of detail of the problem schema (i.e., instantiate it in the new problem.
5. Refine the partial solution.

Abstractional analogy addresses several problems in case-based reasoning and learning. First is the problem of deciding what to transfer from a previous case to a new one and which modifications to

make. Should the previous answer be transferred or should the method of obtaining it be used to find an answer to the new problem? At what level of abstraction should the case-based inference be made? Abstractional analogy solves both of these problems. The mapping done in the first step determines the level of abstraction of the case-based inference. In short, the inference is at the level of detail or abstraction of the commonalities of the two problem descriptions. The method for creating a solution schema determines whether the previous answer or the method of obtaining it should be transferred. In order for abstractional analogy to work, a problem representation must include the set of problem solving steps done to solve the previous problem and the justifications for using those steps. Abstractional analogy creates a solution schema by checking to see if the justifications for each step of the previous problem solving hold in the new problem. If they do, the previous answer can be transferred. If not, the method might be transferred and transformed to fit the specifics of the new case.

The other problem addressed by abstractional analogy is the relationship between problem solving and learning. Learning in other case-based methods happens by indexing cases appropriately so that they can be recalled at a later time. There is thus little generalization done from cases (except when cases are similar to each other). Abstractional analogy creates generalizations in the course of problem solving. Those generalizations are then installed in memory along with the cases. In later problem solving, both the installed generalizations and previous cases are available for case-based reasoning. The memory is responsible for finding the most applicable of those for a new problem. Whether a generalization or a case is recalled, the same set of steps listed above is used to solve the new problem.

This work is reported in Hong Shinn's Ph.D. thesis, entitled "A Unified Approach to Analogical Reasoning." It is implemented in a program called JULIANA, a close cousin to JULIA.

Case-based design: Designers and scholars of design have pointed out that much of design involves combination and reuse of old designs. We have approached meal-planning in exactly this way: as a design problem in which new designs (meals) are created by merging pieces of old designs. Our emphasis has been on the functional pieces of an architecture that allow this to happen. Most case-based reasoning work has concerned itself with the use of one case to solve a new problem. Almost no work in case-based reasoning has concerned itself with the problems involved when a problem must be solved in parts.

The architecture we have proposed has several parts:

- A case-based reasoner retrieves cases similar to the new situation. Cases serve several functions:
  - They suggest whole or partial solutions to problems.
  - They warn of the potential for failure.

  The case-based reasoner uses cases for both of these functions and calls the adapter (below) to adapt previous solutions to fit the new problem.
- A problem decomposer knows how to decompose a problem into component parts. It gets its guidelines from two places: general-purpose decomposition heuristics and previous cases. Decomposition can happen in two ways: by components or by constraints.

- A constraint propagator manages the interactions between parts of a problem. Design problems are at best nearly decomposable. That is, to solve any subpart of a problem, the constraints put on it by other parts of the problem must be taken into account. The constraint propagator propagates the effects of decisions to other parts of the problem. It thus aids with problem recomposition.
- A reason maintenance system keeps track of dependencies and justifications for decisions and notices when constraints are violated.
- An adapter provides several capabilities:
  - It adapts the solutions from previous cases to fit the new situation.
  - It adapts partial solutions to recover from inconsistencies discovered by the reason maintenance system. In this capacity, it too plays a role in problem recomposition.
- A goal scheduler schedules subgoals for execution. Goals can be entered into the scheduler's list by the problem decomposer, the case-based reasoner (when it needs adaptation done), or the reason maintenance system (when it notices an inconsistency). Solving a problem as a whole (through case-based reasoning) is always preferred over decomposition. Adaptation to remove inconsistencies is always preferred over backtracking. The constraint propagator works as an automatic process, and therefore doesn't need scheduling. Whenever something is added to the problem specification or partial solution, the constraint propagator automatically propagates its effects.

The effect of combining this set of processes is that underconstrained problems can be solved by a combination of early and late commitment strategies. The case-based reasoner biases the problem solver to make assumptions or educated guesses about missing information (immediate-commitment) based on its previous experiences, while the constraint propagator gives it the capability of postponing commitment to an answer until more information is available (least-commitment). The goal scheduler imposes top-down control via problem reduction, the constraint propagator contributes bottom-up control by triggering inferences from new information, and the case-based reasoner avoids failure by analyzing remindings from previous cases and makes suggestions that shortcut the work of the other processes. These forms of control enable JULIA to exhibit both reactive and opportunistic behavior. In particular, the combination of top-down and bottom-up control permits opportunistic inference, and the combination of value-driven and reminding-driven control permits a kind of pattern-directed or reactive inference.

JULIA's architecture allows it to solve design problems in a way that is more innovative than many other AI design systems. However, JULIA does not yet do creative design. That is, it does not know how to apply its adaptation heuristics creatively, nor does it have processes for creatively exploring its knowledge base. JULIA does, however, provide a framework for building a creative design system. Such a system will need at least the capabilities that JULIA has.

This work is written up in Tom Hinrich's thesis, "Problem Solving in Open Worlds: A Case Study in Design."

Publications and major addresses: Listed below are publications arising from this project. All conference publications also included presentations. In addition to the publications, there have

been several major addresses resulting from this work. Janet Kolodner gave the keynote address at the Second Annual DARPA Case-Based Reasoning Workshop, entitled "Case-Based Reasoning in Complex Real-World Problem Solving Situations" in May, 1989. She taught several tutorials on Case-Based Reasoning at national and international AI conferences. She presented a keynote address entitled "Contributions of Case-Based Reasoning to Understanding and Improving Human Judgement" at the Conference of the Judgement and Decision Making Society in November, 1989. In August, 1988, she was a participant in the AAAI workshop on Scaling Up Knowledge Bases. The talk given there was based on work supported by this grant. In June, 1987, she was on a panel about new planning paradigms at the Annual Workshop on Conceptual Information Processing. She presented problems with the old planning paradigm and proposed case-based reasoning as a way of dealing with many problematic issues that had not been addressed previously. During Fall, 1987, Janet Kolodner presented research done on JULIA at GTE Labs, Princeton University (Cognitive Science Program), Rutgers University (Computer Science Colloquium Series), Thinking Machines, Inc., MIT (AI Lab), and Tufts/New England Medical Center (Clinical Decision Making Group). During Spring, 1988, she presented talks about case retrieval at Brandeis University, MIT, Tufts/New England Medical Center, GE Research Labs and U. Mass Amherst. Students gave demos of programs supported under this grant at both the First and Second DARPA Case-Based Reasoning Workshops in May, 1988 and May, 1989.

Four Ph.D. theses have resulted from this work:

- Roy Turner: A Schema-Based Model of Adaptive Problem Solving. December, 1989.
- Hong Shinn: A Unified Approach to Analogical Reasoning. December, 1989.
- Elise Turner (nee Hill): Integrating Intention and Convention to Organize Problem Solving Dialogues. December, 1989.
- Tom Hinrichs: Problem Solving in Open Worlds: A Case Study in Design. August, 1991.

Listed below are publications arising from this research:

1. Cullingford, R.E. and Kolodner, J.L. (1986). Interactive Advice Giving. *Proceedings of the IEEE International Conference on Systems, Man & Cybernetics*, Atlanta, Georgia. October. Also, Technical Report GIT-ICS-86/20. School of Information and Computer Science. Georgia Institute of Technology. Atlanta, Georgia. May.
2. Cullingford, R. E. and Graves, M. (1988). Knowledge Acquisition for Natural Language Processing: Two Prospects. *Proceedings of the 1987 Natural Language Planning Workshop on Planning for Future Research: Directions for the Next Decade.* Northeast AI Consortium, Minnowbrook.
3. Graves, M. (1988). Automatic Lexicon Acquisition Using Machine Readable Dictionaries. *Proceedings of the 26th Annual Southeast Regional Conference of the ACM.* Mobile, Alabama. April, 1988.
4. Hill (now Turner), E. (1987). Using Conversational MOPs to Guide Cooperative Problem Solving Dialogues. Unpublished doctoral dissertation proposal. School of Information and Computer Science. Georgia Institute of Technology. Atlanta, GA (Dec.)
5. Hinrichs, T. R. (1988). Towards an architecture for open-world problem solving. *Proceedings of the DARPA Workshop on Case-Based Reasoning.* Clearwater Beach, FL (May).

6. Hinrichs, T. (1988) Problem Solving in Open Worlds: Integrating Case-Based Reasoning and Constraint Propagation. Unpublished doctoral dissertation proposal. School of Information and Computer Science. Georgia Institute of Technology. Atlanta, GA (Feb.)

7. Hinrichs, T. R. (1989). Strategies for Adaptation and Recovery in a Design Problem Solver. *Proceedings of the 1989 DARPA Workshop on Case-Based Reasoning* Pensacola Beach, FL, May, pp. 115 – 118.

8. Hinrichs, T. R. and Kolodner, J. L. (1991). The Roles of Adaptation in Case-Based Design *Proceedings of AAAI-91.* Anaheim, July, 1991.

9. Hinrichs, T. R. and Kolodner, J. L. (1991). The Roles of Adaptation in Case-Based Design (extended version). *Proceedings of the 3rd DARPA Workshop on Case-Based Reasoning.* Washington, DC, May, 1991

10. Kolodner, J. L. (1987). Capitalizing on Failure Through Case-Based Inference. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society.* Seattle, Washington (July).

11. Kolodner, J. L. (1987). Extending Problem Solver Capabilities Through Case-Based Inference. *Proceedings of the Fourth International Machine Learning Workshop.* Irvine, CA (June).

12. Kolodner, J.L. (1987). *Case-Based Inference: A Collection of Papers.* Research Report GIT-ICS 87/18. School of Information and Computer Science. Georgia Institute of Technology. Atlanta, Georgia. April.

13. Kolodner, J. L. (1988). Retrieving Events from a Case Memory: A Parallel Implementation. *Proceedings of the DARPA Workshop on Case-Based Reasoning.* Clearwater Beach, FL (May).

14. Kolodner, J. L. (1989). Selecting the Best Case for a Case-Based Reasoner. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* Ann Arbor, MI, August.

15. Kolodner, J. L. (1989). Judging Which is the Best Case for a Case-Based Reasoner. *Proceedings of the Second Annual DARPA Case-Based Reasoning Workshop,* Pensacola Beach, FL, May.

16. Kolodner, J. L. and Simpson, R. L. (1989). The MEDIATOR: Analysis of an Early Case-Based Reasoner. *Cognitive Science.*

17. Kolodner, J. L. (1991). Improving Human Decision Making Through Case-Based Decision Aiding. *AI Magazine.*

18. Kolodner, J.L. and Thau, R. (1988). *Design and Implementation of a Case Memory.* Thinking Machines, Inc., Technical Report. October, 1988. Also Technical Report GIT-ICS 88/34. School of Information and Computer Science. Georgia Institute of Technology. Atlanta, Georgia. October, 1988.

19. Shinn, H. (1988). Abstractional Analogy: A Model of Analogical Reasoning. *Proceedings of the DARPA Workshop on Case-Based Reasoning.* Clearwater Beach, FL (May).

20. Shinn, H. (1988). The Role of Mapping in Analogical Transfer. *Proceedings of the 1988 Conference of the Cognitive Science Society,* Montreal, Canada, August.

21. Turner, E. H. (1989). Using Dynamic Memory to Interpret Indirect Speech Acts. *Proceedings of the DARPA Workshop on Case-based Reasoning,* pp. 338-340, Pensacola Beach, Florida, May.

22. Turner, E. H. and Cullingford, R. E. (1988). Using Conversation MOPs to Integrate Intention and Convention in Natural Language Processing. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, August, pp. 104 - 110.

23. Turner, E. H. and Cullingford, R. E. (1988). *Using Conversation MOPs in Natural Language Interfaces.* Technical Report No. GIT-ICS-88/14. School of Information and Computer Science. Georgia Institute of Technology, Atlanta, GA.

24. Turner, E. H. and Cullingford, R.E. (1989). Making Conversation Flexible, *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, August.

25. Turner, E. H. and Cullingford, R. E. (1989). Using Conversation MOPs in Natural Language Interfaces. *Discourse Processes*, Vol 12, pp. 63-90.

26. Turner, E. H., 1990. "Planning Conversation: A MOP-based Approach", Technical Report No. 90–67, Computer Science Department, University of New Hampshire. (submitted to AAAI-90).

27. Turner. E.H., "Organizing Dialogue from an Incoherent Stream of Goals", submitted to COLING-92

28. Turner, E. H. and R. E. Cullingford "Exploting Knowledge of Convention to Organize Commuication Goals," to be submitted to *Computational Linguistics.*

29. Turner, R. (1987). Integrating the Use of Previous Problem-solving Experience into Medical Diagnosis. Unpublished doctoral dissertation proposal. School of Information and Computer Science. Georgia Institute of Technology. Atlanta, GA (Dec.)

30. Turner, R. (1988). Organizing and using schematic knowledge for medical diagnosis. *Proceedings of the DARPA Workshop on Case-Based Reasoning.* Clearwater Beach, FL (May).

31. Turner, R.M. (1988). Using schemata for diagnosis. *Proceedings of the Twelfth Annual Symposium on Computer Applications in Medical Care*, Washington, D.C.

32. Turner, R.M. (1988). Opportunistic use of schemata for medical diagnosis. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, Montreal, Canada, August.

33. Turner, R.M. (1989). When reactive planning is not enough: Using contextual schemas to react appropriately to environmental change. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, August.

34. Turner, R.M. (1989). Case-based and schema-based reasoning for problem solving. *Proceedings of the Second DARPA Case-Based Reasoning Workshop*, Pensacola, Florida, May.

35. Turner, R.M. (1989). Using schemas for diagnosis. *Computer Methods and Programs in Biomedicine*, Vol. 30, pp. 199–208.

36. Turner, R. M. (1990). "A Mechanism for Context-sensitive Reasoning", Technical Report No. 90-68. "Department of Computer Science, University of New Hampshire", "Durham, NH 03824".